Prompt Processing

Kian-Tat Lim

Where we left off

- Prompt Processing created a local Butler repo and executed a pipeline (once) on Google Cloud.
- Cloud Run was used to start workers on receipt of next_visit via built-in PubSub-to-webhook adapter.
- Cloud Storage was used to receive FITS images and post notifications via PubSub.
- APDB was implemented on Cloud SQL Postgres.
- Nothing was done with output datasets.

Running LATISS PP on GCP

- With cooperation from TonyJ (installation of gsutil and/or minio client), transfer images directly to GCP and/or USDF from TTS (and then Summit)
- Need to check defining visits on ingest
- If local Butler is thrown away and regenerated on every visit, problems with dimension collisions/skymaps can I think be worked around, but this makes the output dataset problem worse
- next_visit events need to be propagated from TTS/Summit to GCP/Cloud Run
  - At a minimum, a Kafka subscription converting to PubSub
  - Could also be Kafka direct to webhook — only reason not to go directly here is worries about failure modes

Outputs

- Metrics:
  - Use `lsst.verify` to post to Sasquatch
- Alerts:
  - Use Alert Generation/Distribution to send via Kafka and add to Alert Database
- PVIs and diffims:
  - Need to put into Delayed Image Object Store Butler for eventual release
  - Should be an invocation of `butler transfer-datasets` (or Python equivalent)

Running LATISS PP on USDF

- Cloud Run → Kubernetes + Ingress + Horizontal Pod Autoscaler
  - Some thought but no code
- Writing to Cloud Storage → MinIO
  - Can be done at LATISS test scale easily
- Cloud Storage notifications → MinIO notifications
  - Should be relatively easy to convert (e.g. to Redis)
- next_visit PubSub → Kafka
  - Same as for GCP
- APDB Cloud SQL Postgres → Cassandra
  - Not required for short term, especially if no diffim

Future work

- Develop a way to change the pipeline YAML executed (based on deployment, operator configuration, and/or next_visit information)
- Optimize performance
    - Reuse local Butler repo information
    - Reduce cold start time
- Scale up (should be trivial)
- Improve local (non-TTS) testability by better simulating camera image delivery